DOCUMENT RESUME

ED 263 602 CS 209 429

AUTHOR Barker, Thomas T.
TITLE Issues in Software Development in Composition.
PUB DATE May 85
NOTE 13p.; Paper presented at the Meeting of the UCLA Conference on "Computers and Writing: New Directions in Teaching and Research" (Los Angeles, CA, May 4-5, 1985).
PUB TYPE Viewpoints (120) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Computer Software; Educational Innovation; Educational Media; Higher Education; Instructional Materials; *Programing; Secondary Education; Teaching Methods; *Writing Instruction
IDENTIFIERS *English Teachers

ABSTRACT

Noting the increase in the number of teachers developing computer software for composition instruction, this paper explores the issues that are shaping the direction of computer assistance in writing instruction. The first half of the paper deals with specific questions teachers must consider as they design software. These are divided into categories for (1) design, including the pedagogical theory underlying the software, the audience and focus of the software, and professional standards; (2) development, including hiring good programers at competitive wages, management strategies, and testing and quality assurance standards; and (3) distribution, including copyright, licensing, distribution methods, and packaging. The second half of the paper explores the broader issues confronting teachers developing software for composition instruction. These include the environment or conditions under which the software will be used, the role that teachers will play in that environment, and the characteristics of software programing as it is likely to be practiced by English teachers. (HTH)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
X This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Thomas T. Barker
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

ED263602

Thomas T. Barker
Department of English
Texas Tech University
Lubbock, TX 79409

ISSUES IN SOFTWARE DEVELOPMENT IN COMPOSITION

Increased computer use in composition, an urgent need for composition software of all types, and a budding distribution system for composition packages have created a growth in software development in composition. The English teachers who develop (design and or write) their own instructional programs face many important issues, the resolution of which may shape the direction of computer-assistance in composition instruction and may even affect the future of our profession. The purpose of this paper is to raise, explain, and document these issues and thus lay the groundwork for further dialogue.

The evidence suggests that many English teachers have taken up software design. The National Council of Teachers of English offers Winter Workshops on instructional programming, and workshops on programming are now offered routinely at the Conference on College Composition and Communication. Special interest groups such as the Assembly on Computers in English of the NCTE encourage writing teachers to design their own courseware. Publications like Collegiate Microcomputer and Computers in Composition welcome articles on homemade programs. Finally, hardware vendors in the student writing market--IBM, the Apple foundation, and Digital Equipment Corporation--all offer equipment grants to support innovative ideas for educational software.

Digital's Special Investment Grant program, with which I am

CS209 429

most familiar, is typical of the kind of grant support offered by hardware vendors. As a company, Digital is naturally dedicated to selling computers. Equipment that they grant to help some professor put his or her ideas into a program must be justifiable from a cost standpoint. Basically they have three criteria for selecting projects: fitness of the project within Digital's overall marketing strategy, proof of technical expertise of the researcher, and marketability of the product. For those projects that meet these criteria Digital will provide the necessary computer equipment and the support of their considerable marketing organization. Clearly English teachers need to know about markets and about technical matters--areas foreign to most of us.

For some English teachers the experience of publishing software will be like that of Edward Glade, who, in "An Adventure in Microcomputer Software Publication by a Mathematics Instructor," summarized his experience in this way: "I've had a good time and feel nothing but goodwill to all involved. It was a positive experience all around." Let us hope that all feel like professor Glade after their first publication of software. On the other hand, the English instructor may find him or herself following the path outlined by Joseph Bourque in "Understanding and Evaluating: The Humanist as Computer Specialist." Bourque felt "a bit sleazy" approaching his department head to get publication credit for software. His essay, a must for all potential software designers in English, depicts the stone wall of naivete and ignorance that unfortunately awaits many of us eager to claim software publication as evidence of professional development.

Questions

Those English teachers now embarking, or thinking of embarking, on a software development project, and especially those eager to use their computer work as evidence of professional

development, are faced with a brave new world of unexplored territory. For convenience, I have devided that territory into two areas of concern: specific questions and broad issues. The questions, which I will discuss first, concern practical matters. These questions fall into 3 categories: design; development; and distribution.

Design

Questions of design are at the heart of all software development, and they require us to make decisions affecting the quality of our product.

What pedagogical theory will underlie the software? Current theories about collaborative writing, process-based instruction, and rhetorical theory make good starting places for software design.

What market will the software be designed for? Will the software be designed for the college market only, or will it try to cross over into high school and writing in the workplace? Will it be designed for the lab, classroom, or dorm (more on this point later)?

Will the software be curriculum-based or market-based? Will designers have "an eye on what sold well last year," or on the demands of curriculums in the college classroom?

Should the profession institute standards for software design and documentation? The government has standards of quality for design and documentation; so do Engineering societies. How long will it be before college English computing

needs them?

Hugh Burns has explored these questions in his essay "Pandora's Chip: Concerns About Quality CAI," where he asks whether "those of us who are developing computer-assisted instruction are doing too little to insure that we aren't unleashing inadequate, perhaps even incompetent, lessons in our various fields." Many researchers in composition share his concerns, especially those who know how limited computers can be. Pioneers in this area, Bruce Petersen, Cynthia Selfe, and Billie Wahlstrom in "Computer-Assisted Instruction and the Writing Process: Questions for Research and Evaluation" have elaborated on the criteria for judging good composition CAI. Their evaluation procedures and questions about software quality make useful reading for anyone considering developing software. Their warnings echo the concern for enlightened design among many educators. In response to this concern, the NCTE Committee on Instructional Technology is beginning to consider the advisability of instituting standards for the all-important design stage of the process.

Development

Questions about development focus on the production stage of the process. These questions require decisions on how to manage software development projects in English departments. Included under this heading are the following considerations:

> How does one find and hire good programmers? If one decides to hire a programmer, what are the qualities to look for? How much should one pay? How can good programmers be lured away from Engineering departments?
>
> How does one insure against exploitation of student

programmers? In industry good programming costs over $25.00 per hour. Students, even the best of them, often get only about $3.50 per hour out of niggardly English department budgets. Is this fair?

What management strategies work best in an English department? I faced this problem in our microlab at Texas Tech; how to best organize programmers and writers who had classes to attend (or teach) and who could not seem to get together for development team meetings?

What standards for testing and quality assurance are appropriate? Since little software has actually been produced in the area of composition we do not yet know how many test sites are adequate, what reporting procedures should be followed.

For help in the area of development one may profitably examine works in software development by data processing managers over the past 20 years. (Software development does not really go back much farther than that.) Werner L. Frank's Critical Issues in Software: A Guide to Software Economics, Strategy, and Profitability offers design information which I have found useful in managing projects at Texas Tech. The IEEE Computer Society has been publishing in the area of software design and management for years. Their publications can give us a starting place in thinking of ourselves as managers.

Distribution

Answering questions concerning the distribution of software requires us to think of ourselves as salesmen. It forces us to clarify our relationship with textbook publishers and our

responsibilities toward the institutions where we teach. To distribute software effectively, we must consider how to get our product to the intended market and how to get our compensation. Some distribution questions include the following:

Should software be published or distributed through "public domain?" There may come a time when software is so common that we will not claim rights to it as we do printed materials. And, too, there is now discussion of a new concept called "shareware:" software designed to be passed around.

Should individual licenses be sold, or should programs be licensed for sites? Many publishers are reluctant to enter the college market for justifiable fears of wholesale copyright violation. Perhaps site licenses, where a lab or school would pay a blanket fee for unlimited copy privileges, would be an answer.

What method of distribution is appropriate: hardware vendor, software vendor, textbook publisher, self-publication, electronic publication? Depending on the future of the market, all of these channels are candidates for publication. Some distributors seem to have a better understanding of the needs of software developers than others. All are uncertain about how to proceed in the student writing market.

How should the software be packaged? This question fascinates me personally. Whole new areas of book and "floppy book" design are being opened up by the possibility of merging textbook with computer manual. On a more immediately practical level, we must decide on pricing and on whether teachers or

students will pay for the programs.

How can we insure that developers get their fair returns? Standards have yet to appear for royalty agreements, especially in areas where lots of money can be made. Additionally, programming is a time-consuming and expensive endeavor. Who pays?

These questions of the ownership and the distribution of composition software are just beginning to surface. I have found the Participate Network from N.Y.I.T., especially the computer conference on "ED WARE," to be one of the few forums for the thorny questions involved here. It was encouraging to see textbook publishers and software developers dialoguing openly at the Conference on Computers and Writing in Los Angeles in May, 1985. At that conference one could clearly see mutual interest and commitment. But there is still far to go in tackling problems in the areas of support for development and maintenance of software after the sale.

These questions of design, development, and distribution differ from the more fundamental issues which, as I see them, do not involve practical decisions. Such issues address the new role of programming and software design discussed below. The two principal issues affecting instructional programmers in English are: 1) the issue of environment; and 2) the issue of the teacher's role in computer-assisted instruction.

Issues

Environment

Environment is a major issue because it forces us to question the logistics of computers in composition instruction. Where will these programs be used? Will they be housed in a lab, in student

rooms, in terminal rooms, in commercial computer rooms? Will students use them alone or in collaborative groups? None of these questions has been answered.

To give an example: consider the recent rise in the numbers of microlabs in English departments and the type of environment these labs create. In labs a number of computers must all run the same programs and must be used by different people, sometimes for an hour at a time, sometimes for 5 hours. Updates of word processing programs and revision aids must be maintained on anywhere from 5 to 30 diskettes. To accommodate this environment, I had to write a special maintenance utility to go along with a program, so that instructors could transfer files easily onto stacks of disks. This utility program would not be necessary if the program were used in a dorm room or office. Additionally, the program is written for a high-turnover environment, characteristic of the microlab. My program has all kinds of built-in compromises for the one-time user which prove to be built-in frustrations for the expert. Even more disturbing is the realization that, by the very design of the program, we assume an environment that may be radically different in 5 years or that may not exist in 5 years.

The Teacher's Role in CAI

Just as designers must make assumptions about the environment for their software, so they must make assumptions about the role of the teacher in that environment. We can begin by asking what our role is now. Most of us are pretty comfortably set in our ways. We tend to see our selves at the center of things: lecturing; directing workshops; leading discussions. Until recently, I saw myself as the "educated" half of the teacher/student relationship. But computers can change our models. While now we think of teachers as channels of information, in the future much of that information may be channeled by computers. We may now think of teachers (and

textbooks) as reference sources for students; this role may be assumed by the computer. We now think of ourselves as encouragers of learning; this role may be assumed by the computer. How much more will the computer involve the instructor as a writer in a collaborative environment?

The role of teacher may become specialized, in fact, it may come to resemble that of a manager or facilitator. The teacher may find him or herself involved more in computer training than before, perhaps learning along with the student. I take on the role of interpreter to many of my students. Sometimes I'm a snoop; I watch how people learn and ask questions like "How did you figure that out?" The presence of computer-assisted instruction has fragmented my once unified classroom/teacher identity. The direction which that fragmentation will take may depend significantly on the work of those who design the software.

English Teachers as Software Designers

What kind of role will we design for our programs and for ourselves? The answer to that question may lie in an understanding of the characteristics of software programming as it is liable to be practiced by English teachers. I have tried to imagine how English teachers would proceed, but the reader should beware that the following assertions are formed from a mixture of experience, observation, and speculation.

* Many instructors find that design and programming offer opportunities for managing software production. Work in microlabs encourages one to explore learning systems, with teachers as education managers.

* English teachers, with relatively limited technical knowledge, have the advantage of working from their imaginations rather than technical limitations. Perhaps

their work will be explorative, learning along with their students.

* English teachers are writers first and programmers second, not the other way around as has been the case with most programs. As communicators and writers they bring a special expertise to the design of screens and on-line documentation.

* English teachers are subject matter experts as well as programmers. They are more serendipitous. They must, however, decide on what content is appropriate for computers to teach and what is best taught by warm human beings.

* English teachers keep up with research sources. What are the appropriate areas for further research by programmers: pedagogy, information transfer, artificial intelligence, database management? Perhaps their programs will have built-in research capabilities.

Conclusion

The characteristics described above suggest some light in an otherwise dim future. But as I think I have shown, with careful planning the designer of instructional software can affect our role as teachers by building in an appropriate teacher involvement. The designer can also help us envision the failities in which we will use computers.

Part of the reason for the barrier among our colleagues to accepting computer-assistance is that they are unenlightened about its potential and that they feel their teaching role threatened. But a greater barrier would be a lack of awareness on the software designer's part. Perhaps we all have much to gain from realizing

our new found power to shape the instructional setting and to define our role in it.

BIBLIOGRAPHY

Joseph H. Bourque. "Understanding and Evaluating: The Humanist as Computer Specialist". College English, 45:1, January, 1983.

Hugh Burns. "Pandora's Chip: Concerns About Quality CAI". Pipeline 6:2, Fall, 1981.

Werner L. Frank. Critical Issues in Software: A Guide to Software Economics, Strategy, and Profitability. New York: John Wiley & Sons, 1983.

Thomas L. Franke. "Computers and Writing Instruction: Issues for Policymakers". Pipeline, 8:2 (Fall, 1983), pp. 23-24.

Glade, Edward H. H. 3rd. "An Adventure in Microcomputer Software Publication by a Mathematics Instructor." Collegiate Microcomputer, 2:3, (August, 1984), pp. 241-244.

Roger A. Grice. "Producing and Using Online Information: The Display Screen as an Extension of the Printed Page" in Proceedings of the 1984 IEEE Professional Communication Society Conference, Atlantic City, 1984: p. 164.

"A New Wave of Professionals Become Programmers," Infoworld, 6:6 (February 6, 1984), pp. 82-84.

Petersen, Bruce T., Cynthia L. Selfe and Billie J. Wahlstrom. "Computer-Assisted Instruction and the Writing Process: Questions for Research and Evaluation." College Composition and Communication, 35:1 (February, 1984), pp. 98-101.